REMARKS

Please reconsider the present application in view of the above amendments and the following remarks. Applicant thanks the Examiner for the courtesies extended in the telephonic Examiner Interview of May 4, 2005. Applicant also thanks the Examiner for withdrawing the rejection of the claims under 35 U.S.C. § 101, and for accepting the formal drawings submitted on July 17, 2001.

Disposition of Claims

Claims 1-17 are pending in the present application. Claims 1, 10, 11, and 15 are independent. The remaining claims depend, directly or indirectly, from claims 1, 11, and 15.

Claim Amendments

Independent claims 1, 10, 11, and 15 have been amended by way of this reply. Specifically, claims 1, 10, 11, and 15 have been amended, as discussed in the telephonic Examiner interview of May 4, 2205, to clarify that an archive file is compressed and based on a structure of an object-oriented class. No new matter has been added by way of these amendments, as support for these amendments may be found, for example, in paragraphs [0094]-[0098] of the present application. Additionally, dependent claims 4 and 5 have been amended to be consistent with amendments made to claim 1, and dependent claims 7, 14, and 21 have been amended to be consistent with claims 6, 13, and 20, respectively. No new matter has been made by way of these amendments.

Rejection(s) under 35 U.S.C § 103

Claims 1, 8, 11, 12, and 15-17

Claims 1, 8, 11, 12, and 15-17 are rejected under 35 U.S.C. § 103(a) as being obvious over U.S. Patent No. 5,920,867 issued to Van Huben *et al.* (hereinafter "Van Huben") in view of U.S. Patent No. 5,201,047 issued to Maki *et al.* (hereinafter "Maki"). Independent claims 1, 11, and 15 have been amended in this reply to clarify the present invention recited. To the extent that this rejection may still apply to the amended claims, the rejection is respectfully traversed.

The present invention is directed to a method of creating and defining custom data fields in a process management system. As discussed with reference to one embodiment of the present invention, a Process Manager (PM) (78) is an application that runs on top of an application server to develop, deploy, and manage business processes (*see, e.g.,* Specification, paragraph [0015]). Process Manager Builder (PMB) (91) is a visual process design tool that can be used to create and deploy PM (78) applications (*see, e.g.,* Specification, paragraph [0017]).

While PMB (91) allows a user to build applications and control the flow of processes, it may occasionally be necessary to tweak applications by going outside the PMB (91). In other words, a data field different from the built-in data sources may be necessary. In such a case, a custom data field may be defined, created, and implemented when building an application (*see e.g.,* Specification, paragraph [0060]).

As seen with respect to Figure 10 of the Specification, creating a custom data field begins with creating a JavaScript Bean (JSB) file (step 140) to specify the visible field properties in the PMB (91). Next, a Java™ class is written to define the custom data field (step 142) in terms of presentation and data management capabilities. If necessary, images to depict the data field in the PMB interface are created (step 144). Then, the JSB and Java™ classes are

packaged into a zip or jar archive (step 146), and a data field is inserted and added to the archive file as a new class in the PMB (91) (step 148). The final step, after creating and defining the custom data field, is deploying and testing the custom data field (step 150) (*see, e.g.,* Specification, paragraph [0063]).

As discussed above, after the object-oriented custom data field classes are compiled, the JSB file and any images to represent the datafield in the PMB (91) are defined. These files are then packaged into a zip or jar archive file based on the structure of the object-oriented class (*see, e.g.,* Specification, paragraph [0093]).

Figure 15 shows an example archive file for a custom data field in accordance with one embodiment of the present invention. Packaging such a custom data field begins by creating an archive file in a utility application such as, for example, WinZip or PKZip (*see, e.g.,* Specification, paragraph [0094]). As shown in Figure 15, each of the files added to the archive is compressed from a first size to a packed size by a ratio that varies depending on various factors that are well understood by one skilled in the art. Further, the directory structure of the archive reflects the package structure of the class. For example, as shown in Figure 15, if the class files are in the package customer.fields, then the class files in the archive file are in the directory customer/fields (200) (*see, e.g.,* Specification, paragraph [0094]).

Accordingly, amended independent claim 1 of the present invention requires packaging a file and a model into an archive file, where the archive file is compressed and based on a structure of an object-oriented class, and inserting the custom data field and adding the archive file into the process management system as the object-oriented class. Amended independent claims 11 and 15 include similar limitations to claim 1.

Van Huben, in contrast to the present invention, completely fails to teach or disclose packaging a file and an archive file into an archive file, where the archive file is compressed and

based on a structure of an object-oriented class. The Examiner asserts that Van Huben teaches packaging a file and a model into an archive file. However, Van Huben does not teach packaging a file and a model into an archive file, where the archive file is compressed and based on a structure of an object-oriented class. The Examiner notes that Van Huben is directed to archiving and backing up data in a library. Van Huben simply describes a simple archival process to a tape drive or another repository. For example, when a design control repository becomes corrupted or an individual data object is deleted, a back-up copy allows the repository or the object to be restored (*see* Van Huben, col. 28, lines 41-62). This is clearly not packaging a file specifying visible field properties and a model of a custom data field into an archive file, where the archive file is compressed and based on a structure of an object-oriented class, as required by amended independent claims 1, 11, and 15.

As discussed above, Van Huben fails to teach or disclose all the limitations of independent claims 1, 11, and 15. Maki also fails to teach or disclose all the limitations of these claims and fails to provide that which Van Huben lacks. As discussed above in reference to Van Huben, Maki also does not teach or disclose packaging a file and a model into an archive file, where the archive file is compressed and based on a structure of an object-oriented class. Further, Maki does not teach or disclose inserting the custom data field and adding the archive file into the process management system as the object-oriented class.

Maki is directed to automated classification for group technology applications. Group technology is based on exploiting similarities in parts or items for classification purposes (*see* Maki, col. 1, lines 6-2). Specifically, Maki describes a codeless classification system for locating items with similar attributes, avoiding necessary reclassification when a code hierarchy changes (*see* Maki, *e.g.*, col. 2, lines 66-68). However, Maki completely fails to contemplate

packaging a file and a model into an archive file, where the archive file is compressed and based on a structure of an object-oriented class.

The Examiner further asserts that Maki teaches "inserting the custom data field," in that Maki teaches classification tree nodes with new attributes for other business entities constructed. However, Maki teaches the use of item classification tree queries, and shows exemplary results of such queries, as shown in Figures 4A-4E of Maki. As discussed with respect to Figure 9B of Maki, Maki teaches that the templates for attributes may be modified, and that the classification system may be implemented using a precompiled query program (*see* Maki, col. 4, lines 23-53). Maki does not show or suggest inserting a custom data field and adding the archive file into the process management system as the object-oriented class. Further, the Examiner acknowledges that Van Huben does not teach inserting the custom data field and adding the archive file into the process management system as a new class as recited in the claims.

Further, in rejecting the claims of the present invention, the Examiner has improperly combined Van Huben with Maki, as Van Huben teaches away from Maki. Within the Van Huben reference, the deficiencies of Maki are spelled out in such a fashion that one skilled in the art would not combine Van Huben with Maki. Specifically, Van Huben states that Maki "requires that the data items being grouped share at least one common attribute to enable the grouping, and therefore fails to address problems of managing data aggregates formed from disparate and unrelated data objects" (*see* Van Huben, col. 5, lines 5-17). Such disparaging comments by Van Huben clearly suggest that the references are not meant to be combined. Accordingly, the Examiner improperly combined Maki with Van Huben in an attempt to render the claimed invention obvious.

In view of the above, Van Huben and Maki, (i) whether taken separately or in combination, fail to show or suggest the present invention as recited in amended independent claims 1, 11, and 15 and (ii) are not properly combinable. Thus, amended independent claims 1, 11, and 15 are patentable over Van Huben and Maki. Dependent claims 8, 12, 16, and 17 are allowable for at least the same reasons. Accordingly, withdrawal of this rejection is respectfully requested.

Claims 4-7, 10, 13, 14, and 18-21

Claims 4-7, 10, 13, 14, and 18-21 are rejected under 35 U.S.C. § 103(a) as being obvious over Van Huben (as cited above) in view of Maki (as cited above), and further in view of Applicant Admitted Prior Art (hereinafter "AAPA"). Independent claims 1, 10, 11, and 18 have been amended in this reply to clarify the present invention recited. To the extent that this rejection may still apply to the amended claims, the rejection is respectfully traversed.

As discussed above, Van Huben and Maki, whether taken separately or in combination, fail to show or suggest all the limitations of independent claims 1, 10, and 11. Amended independent claim 18 includes limitations similar to claim 1, and is allowable for at least the above reasons.

Further, AAPA fails to teach or disclose that which Van Huben and Maki lack. AAPA does not teach or suggest the present invention as recited by amended independent claims 1, 10, 11, and 18. AAPA merely discloses a universal platform (Java™) for providing resources to various platforms across the Internet (*see, e.g.,* Specification, paragraphs [0002]-[0003]) and related applications. For example, AAPA discusses a process manager (78), which allows a user to create web-based applications, and a process manager builder, which allows a user to create and PM applications. However, AAPA does not show or suggest packaging a file and a model into an archive file, where the archive file is compressed and based on a structure of an object-

oriented class. Further, AAPA does not show or suggest inserting a custom data field and adding the archive file into the process management system as the object-oriented class.

Further, the Examiner acknowledges that Van Huben in view of Maki fails to show or suggest a model of a custom data field as required by claims 6 and 13 of the present invention. However, the Examiner asserts that AAPA teaches that a model comprises a written class and at least two implemented interfaces. Applicant respectfully disagrees because AAPA does not teach or suggest a model of a custom data field as recited in claims 6 and 13. Specifically, the Examiner asserts that the process map shows the model in Figures 5 and 8. However, Figures 5 and 8 of the present application show screen shots of a visual process map (120) in a Process Manager Builder (91), and a Process Manager Administrator (94), respectively. A visual process map (120) allows a user to graphically modify a process (*see, e.g.,* Specification, paragraph [0018]). A Process Manager Administrator (94) is merely an administrative interface designed to: (a) manage Process Manager (PM) clusters (96) and (b) monitor and manage deployed processes or applications (*see, e.g.,* Specification, *e.g.,* paragraph [0023]). Thus, AAPA clearly fails to show or suggest that a model comprises a written class and at least two implemented interfaces. In fact, Figures 5 and 8 fail to show or suggest *anything* associated with a model.

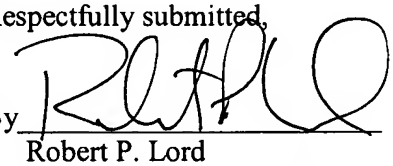
In view of the above, Van Huben, Maki, and AAPA, whether taken separately or in combination, fail to show or suggest the present invention as recited in amended independent claims 1, 10, 11, and 18. Thus, amended independent claims 1, 10, 11, and 18 are patentable over Van Huben, Maki, and AAPA. Claims 4-7, 13, 14, and 19-21, which depend directly or indirectly from claims 1, 10, 11, and 18, are allowable for at least the same reasons. Accordingly, withdrawal of this rejection is respectfully requested.

Conclusion

Applicant believes this reply is fully responsive to all outstanding issues and places the present application in condition for allowance. If this belief is incorrect, or other issues arise, the Examiner is encouraged to contact the undersigned or his associates at the telephone number listed below. Please apply any charges not covered, or any credits, to Deposit Account 50-0591 (Reference Number 13220.002001; P5653).

Dated: June 23, 2005

Respectfully submitted,

By 

Robert P. Lord
Registration No.: 46,479
OSHA · LIANG LLP
1221 McKinney St., Suite 2800
Houston, Texas 77010
(713) 228-8600
(713) 228-8778 (Fax)
Attorney for Applicant

103574_1